Projekt

Teoria i metody optymalizacji

Harmony Search

Autor:
Artur Gasiński, 218685
Bartosz Lenartowicz, 218518

Termin: wt 9:15-11:00

Prowadzący:
Dr inż. Ewa Szlachcic

26 maja 2018

Spis treści

1 Wstęp **2**

2 Opis działania algorytmu **2**

3 Implementacja **2**
3.1 GUI . . . 3
3.1.1 Parametry . . . 4
3.1.2 Wykres . . . 4
3.1.3 Tabela Rozwiązań . . . 4
3.2 Pliki . . . 4
3.2.1 Katalog `contants` . . . 5
3.2.2 Katalog `core` . . . 5
3.2.3 Katalog `gui` . . . 6
3.2.4 Katalog `helpers` . . . 7

4 Przykładowy rozwiązań oraz testy **7**
4.1 Funkcja czterech minimum . . . 7
4.1.1 Zmiana zakresu dla argumentów . . . 7
4.2 Funkcja Gemm . . . 9
4.2.1 Zmiana zakresu dla argumentów . . . 9
4.3 Funkcja Himmelblau . . . 10
4.4 Funkcja trzech zmiennych . . . 10
4.5 Funkcja ceny złota . . . 11
4.6 Dwuwymiarowa funkcja sinusoidalna . . . 11
4.7 Dwuwymiarowa funkcja sinusoidalna z eksponentom . . . 11
4.8 Podsumowanie testów . . . 12

5 Problemy podczas pracy **12**

6 Wnioski końcowe **12**

Bibilografia **13**

1 Wstęp

Większość metod optymalizacji procesów produkcyjnych czy logistycznych sprowadza się do przedstawienia zależności pomiędzy procesami za pomocą funkcji. Pod postacią zmiennych tego równania ukazane są czynności, którymi jest się w stanie manipulować. Takie równanie nosi nazwę funkcji celu. By właściwie zoptymalizować proces należy znaleźć, w zależności co poddane jest optymalizacji, minimum bądź maksimum (najlepiej globalne) takiej funkcji. Obliczanie minimalnej bądź maksymalnej wartości, zależnej od wielu zmiennych w pożądanym czasie, nie jest zadaniem łatwym. Istnieją specjalne algorytmy do rozwiązywania zadań tego typu. Niniejsza praca została napisana by przedstawić jeden z takich algorytmów – Harmony Search.

Wpierw w rozdziale 2 został przedstawiony opis tego algorytmu. Rozdział 3 przedstawia szczegółowo napisany program komputerowy działający na bazie tego algorytmu. Następnie w rozdziale 4 zostały przedstawione przykłady rozwiązań obliczonych przez program. W rozdziale 5 przedstawiono problemy z jakimi borykano się podczas tworzenia programu. Rozdziałem **??** zamykającym pracę jest rozdział przedstawiający podsumowanie pracy nad algorytmem.

2 Opis działania algorytmu

Algorytm Harmony Search został przedstawiony w 2001 przez Zong Woo Geem, Joong Hoon Kim oraz G.V. Loganathan w pracy Ą New Heuristic Optimization Algorithm: Harmony Search" [5]. Inspiracją dla algorytmu było szukanie przez muzyków jazzowych, podczas improwizacji, najlepszych harmonii dźwięków. Jego matematyczny odpowiednik umożliwia znajdowanie minimów lokalnych funkcji wielu zmiennych. Główna zasada polega wyszukiwaniu minimum na podstawie wartości wcześniej obliczonych, oraz szukaniu nowych wartości zmiennych z góry określonym prawdopodobieństwem. Jednym z ważniejszych elementów algorytmu jest pamięć harmonii `HM` (*ang. harmony memory size*). Jest to zbiór rozwiązań funkcji f wraz z wartościami zmiennych x. Schemat tablicy `HM` umieszczony jest na rysunku 1. Wielkość pamięci m deklarowana jest z góry i podczas kolejnych iteracji nie zmienia rozmiaru. Algorytm wymaga by na początku działania określić przedział wartości dla każdej zmiennej $x \in < a, b >$. Początkowe wartości tablicy `HM` są losowane z całego przedziału. Gdy tablica zostanie zapełniona początkowymi wartościami rozpoczyna się część iteracyjna algorytmu. Polega na wyszukiwaniu kolejnego rozwiązania poprzez wyznaczanie zmiennych wedle dwóch prawdopodobieństw. Pierwsze `HMCR` (*ang. harmony memory consideration ratio*) określa współczynnik doboru tonu z pamięci, drugie `PAR` (*ang. pitch adjustment ratio*) nazywane jest współczynnikiem dostosowania tonu. Oczywistym jest, że przedział wartości prawdopodobieństw określany jest na $< 0, 1 >$. Istnieją trzy reguły wyboru zmiennych. Pierwsza z nich mówi, że z prawdopodobieństwem $P_1 =$ `HMCR` wartość nowej zmiennej xn wybierana jest z pośród zmiennych zapamiętanych w tablicy `HM`. Druga, że wartością prawdopodobieństwa $P_2 = (1 -$ `HMCR`$)$ wartość zmiennej losowana jest z całego przedziału $< a, b >$, a zasada trzecia, że zmienna ustalona w regule pierwszej modyfikowana jest z prawdopodobieństwem $P_3 =$ (`HMCR` $*$ `PAR`). W praktyce sprowadza się to do dodania wylosowanej zmiennej x z reguły pierwszej liczby z przedziału $< -b, b >$. Następnie gdy zostaną wylosowane wszystkie wartości x_n obliczana jest dla nich wartość funkcji f. Następnie jest ona porównywana z wartościami umieszczonymi w tablicy `HM`. Jeżeli jest mniejsza niż największa wartość z tablicy, zamienia się ją z największą wartością. Na tym kończy się jedna iteracja. Następnie wykonuje się kolejne iteracje do momentu znalezienia rozwiązania $f \pm \epsilon$ lub do osiągnięcia maksymalnej ilości iteracji. Polski opis algorytmu można również znaleźć np. w pracy [3].

x_{11}	$\cdots$	x_{n1}	$= f_1$
$\vdots$	$\ddots$	$\vdots$	$\vdots$
x_{1m}	$\cdots$	x_{nm}	$= f_m$

Rysunek 1: Schemat tabeli *Harmony Memory*

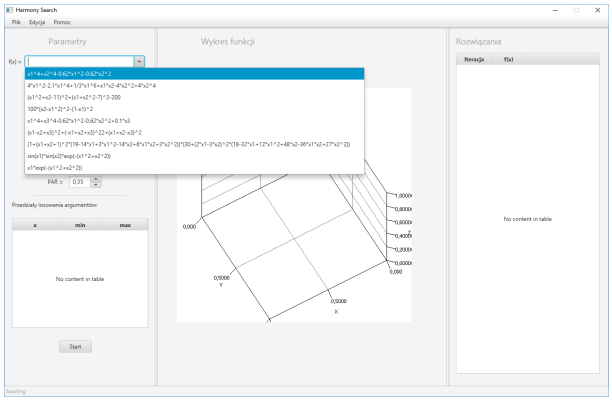
3 Implementacja

Program do obliczania minimum funkcji na podstawie algorytmu Harmony Search został napisany w języku Java. Technologią dostarczającą GUI dla użytkownika była *JavaFx*. Korzystał on również z zewnętrznej biblioteki do obliczania wartości funkcji w punkcie [4]. Wykres prezentujący najlepsze

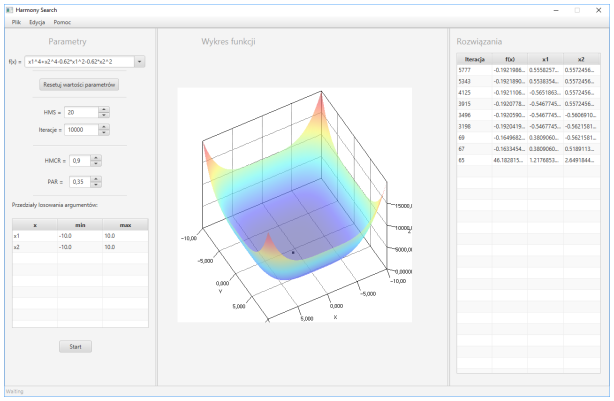
rozwiązanie zaznaczone na wykroju z przestrzeni rozwiązań funkcji został stworzony za pomocą biblioteki *jzy3D*. Poniżej zostanie szczegółowo omówiony wygląd czyli `GUI` programu oraz struktura plików i katalogów.

3.1 GUI

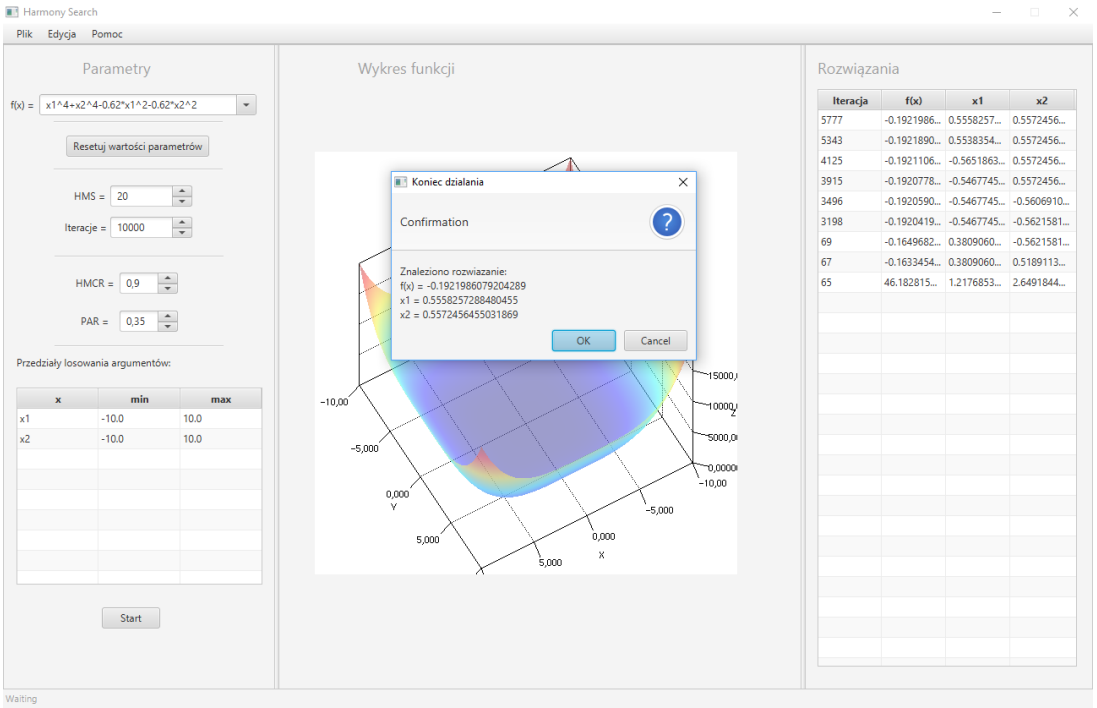
By ułatwić wprowadzanie funkcji do programu oraz bezproblemowo odczytywać wyniki obliczone przez program zostało stworzone GUI programu. Na rysunkach 2, 3 i 4 przedstawiono etapy działania programu. Poszczególne elementy GUI podczas działania programu nie zmieniają swojego położenia. Jedynie pustych w miejscach zostają wyświetlone ich parametry, dlatego by omówić elementy posłużono się rysunkiem 4, gdzie widnieje obraz programu po obliczeniach. Główne okno programu zostało podzielone pionowo na trzy części. Pierwsza część od lewej przeznaczona jest do wpisania funkcji i jej parametrów, środkowa prezentuje wykres, a część prawa w tabelce wyniki obliczeń programu. Wszystkie części zostaną szczegółowo opisane poniżej.



Rysunek 2: Wpisywanie funkcji



Rysunek 3: Wykres z zaznaczonym rozwiązaniem



Rysunek 4: Obliczone rozwiązanie wraz z komunikatem

3.1.1 Parametry

W części parametrów można dostrzec, że zaraz pod paskiem edycji znajduje miejsce do wpisania funkcji. Jest to ta sama funkcja której minimum należy wyznaczyć. Okno umożliwia również wybór takiej funkcji z wcześniej zadeklarowanych w programie. Dokonuje się tego klikając w rozwijane menu. Zostało to przedstawione na rysunku 2. Po wybraniu funkcji automatycznie zostaną domyślnie uzupełnione wszystkie parametry obliczeń. Oczywiście możliwa jest dostosowanie algorytmu do własnych potrzeb. Poniżej okna funkcji znajduje się przycisk „`Resetuj wartości parametrów`" umożliwiający, jak mówi nazwa, reset do parametrów domyślnych. Poniżej przycisku zostały umieszczone po sobie cztery główne parametry algorytmu *Harmony Search*.
Pierwszy z parametrów `HMS` (*ang. harmony memory size*) reguluje ilość zapamiętywanych rozwiązań podczas wyszukiwania. Domyślnie rozmiar jest zadeklarowany na 20. Poniżej znajduje się okno o nazwie `iteracje`. Domyślnie jest zadeklarowana maksymalna wartość iteracji równa 10000. W każdej iteracji programu algorytm wyszukuje nowe rozwiązanie i sprawdza je z tymi zapisanymi w `HMS`. Czym większa ilość iteracji tym algorytm powinien obliczać dokładniejsze rozwiązanie, kosztem wydłużonego czasu obliczeń. Następnie pod polem `iteracje` znajduje się pole `HMCR` (*ang. harmony memory consideration ratio*). Tłumaczenie określa go jako współczynnik wyboru tonu z pamięci. Jest to współczynnik prawdopodobieństwa wybierany z zakresu $(0, 1)$. Czym `HMCR` jest większe tym wyszukiwane wartości x następnych rozwiązań będą zbliżone do tych istniejących już w tablicy `HMS`. Poniżej znajduje się drugi parametr prawdopodobieństwa dla funkcji o nazwie `PAR` *(ang. pitch adjustment ratio)* tłumaczony jako współczynnik dostosowywania tonu. Wybierany jest również z zakresu $(0, 1)$. Na samym dole kolumny znajduje się tabela `Przedziały losowania argumentów`. W tabeli deklarowany jest przedziały wszystkich wartości x dla których ma być poszukiwane rozwiązanie. Domyślnie parametry dla wszystkich x brane są z przedziału $(-10, 10)$. Algorytm z dużą dokładnością znajdzie najmniejsze minimum lokalne dla funkcji z podanego przedziału. Po wybraniu wszystkich parametrów można przystąpić do uruchomienia algorytmu przyciskiem `Start` znajdującym się na na dole kolumny.

3.1.2 Wykres

Środkowy obszar okna głównego przeznaczony jest do wyświetlania wykresu funkcji. Jak wspomniane zostało na początku rozdziału 3, do rysowania wykresów została użyta zewnętrzna biblioteka *jzy3D*. Umożliwia ona tworzenie wykresów dwu i trzy wymiarowych. Wykresy można obracać w dowolny sposób. Oś X i Y wykresu wyskalowana jest do najmniejszego i największego parametru wszystkich wartość x. Dodatkowo wartości zostały pokolorowane w tak, że mniejsze wartości funkcji mają kolory zimniejsze, a wyższe kolory cieplejsze, analogicznie do tego jak przedstawia się wysokość n.p.m. w kartografii. Dodatkowo czarną kropką zaznaczona została najmniejsza obliczona wartość funkcji.

3.1.3 Tabela Rozwiązań

Po prawej stronie okna głównego można dostrzec najlepsze rozwiązania wyszukane przez program. Gdy znajdowane jest rozwiązanie lepsze od najlepszego rozwiązania zapamiętanego w `HMS` dodawane jest ono jako pierwszy wiersz w tabeli. Poprzednie rozwiązania przesuwane są o jedno miejsce niżej. Pozwala to śledzić jaka wartości jest aktualnie największa oraz od razu ukazuje się obok numer iteracji tego rozwiązania. Gdy algorytm długo oblicza najlepszy wynik okno pozwala śledzić postęp obliczeń. Dodatkowo po wykonaniu wszystkich iteracji zostaje wyświetlony komunikat z najlepszym rozwiązaniem (pierwszym od góry), co przedstawia rysunek 4.

3.2 Pliki

Program był pisany w środowisku programistycznym *IntelliJ IDEA*. Jak wcześniej wspomniano do stworzenia GUI programu użyto technologii *JavaFx* umożliwiającej tworzenie aplikacji okienkowych. Dodatkowo by ułatwić zarządzanie bibliotekami wewnątrz programu użyto wsparcia ze strony *Maven*. By ułatwić zrozumienie tekstu stworzono graficzną wersję struktury plików i katalogów. Główny kod programu został umieszczony katalogu `src`. Katalog zawiera w sobie dwa inne katalogi `resourses` oraz `java`. Katalog `resourses` posiada jeden folder w którym umieszczony jest plik `main.fxml` odpowiedzialny GUI programu. W tym pliku zapisane są informacje o ułożeniu i wielkości elementów, które widzi użytkownik. Można powiedzieć że plik odpowiada za foreground programu.
Cała background programu czyli jego mechanika jest umieszczona w folderze `java`. Posiada on dwa podfoldery `org.jzy3d` oraz `com.blag.harmonysearch`. Pierwszy katalog odpowiedzialny jest za generowanie wykresów w oknie opisywanym w rozdziale 3.1.2. Są to pliki zewnętrzne pobrane ze strony [2]. Katalog

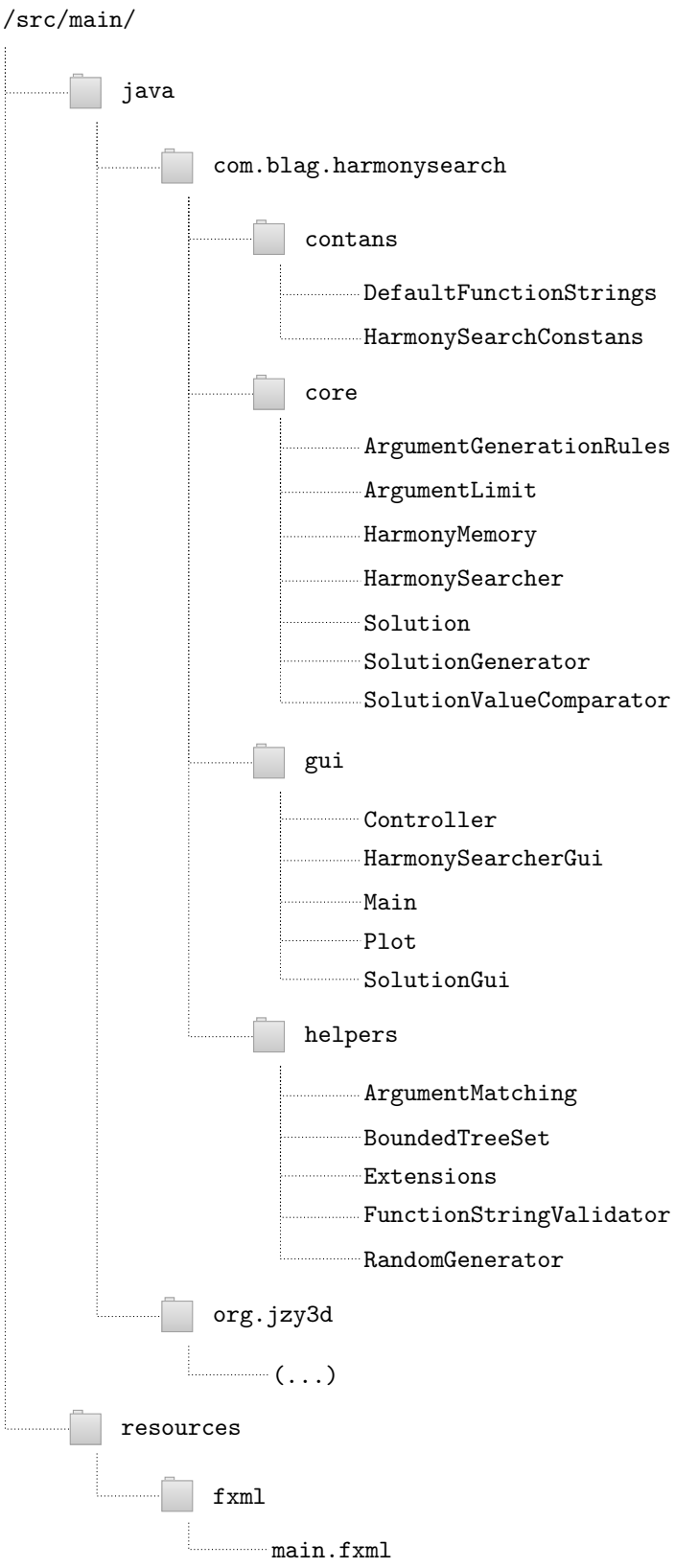
`com.blag.harmonysearch` zawiera cztery foldery. Każdy z nich posiada pliki napisane na potrzeby tego projektu. Dla czytelności opisu zostaną one przedstawione jako kolejne podrozdziały.

3.2.1 Katalog `contants`

W tym katalogu zadeklarowane są tylko dwa pliki. W pliku `DefaultFunctionStrings` zadeklarowano przykładowe funkcję wspominane w 3.1.1. Plik `HarmonySearchConstans` odpowiada za domyślne parametry.

3.2.2 Katalog `core`

W katalogu `core` znajdują się najważniejsze pliki odpowiedzialne za działanie programu. Klasa `ArgumentGenerationRules` jest klasą przechowującą zasady generowania argumentów. Klasa `ArgumentLimit` implementuje zakres z jakiego wyszukiwane będą elementy. Główną klasą gdzie przechowywane są wartości jest klasa `HarmonyMemory`. Posiada ona tablice rozwiązań, której wielkość jest regulowana z GUI programu. Tablica w została zaimplementowana jako `SortedSet` co ogromnie przyspiesza przeszukiwanie. Klasą gdzie zadeklarowany jest główny algorytm czyli gdzie wykonywane są operacje obliczeń jest klasa `HarmonySearcher`. Wylosowane rozwiązanie przekazywane jest jako lista złożona z typu `Solution`. Klasa `SolutionGenerator` umożliwia wygenerowanie nowych wartości x dla którego zostanie obliczone rozwiązanie. Ostatnia klasa w tym pliku `SolutionValueComparator`, porównuje czy rozwiązanie dla wylosowanych x jest lepsze od rozwiązań zadeklarowanych w tablicy.



3.2.3 Katalog `gui`

W katalogu znajdują się klasy odpowiedzialne za kontrolowanie tego co ma wypisać bądź wyrysować się na ekranie. Katalog `Controller` dostarcza bezpośrednio wartości do pliku `main.fxml`. Do obliczeń pobiera wartości z klasy `HarmonySearcherGui`. Klasa `Main` aktywowana jest po rozpoczęciu pracy pro-

gramu jako pierwsza i pobudza całość programu do działania. Wykres tworzony jest dzięki obiektowi klasy `plot`, a w obiekcie klasy `SolitionGui` wyświetlane są wyniki. Klasy `Gui` są rozszerzeniem klas z folderu `core` opisanych w 3.2.2.

3.2.4 Katalog `helpers`

Katalog zawiera głównie klasy pomocne będące rozszerzeniem klas podstawowych *Javy*. Klasy rozwijają Listy, drzewa, generatory i wiele innych klas o funkcjonalności niezbędne do działania programu. Zostały wykorzystane jako wsparcie w implementacji w plikach z katalogu `core`.

4 Przykładowy rozwiązań oraz testy

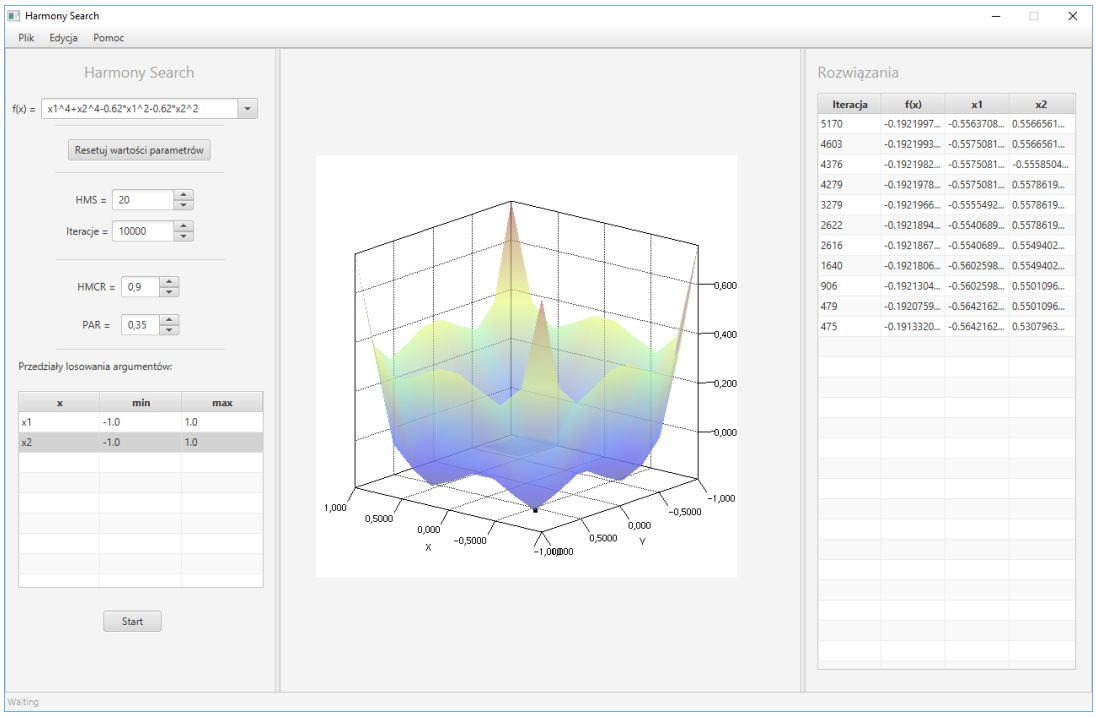
W tej części zostały przedstawione trzy przykłady funkcji dwóch zmiennych x_1 i x_2 wraz z obliczonymi rozwiązaniami.

4.1 Funkcja czterech minimum

Pierwsza zostanie przedstawiona funkcja posiadająca cztery minima lokalne. Posiada ona następujący wzór:

$$f(x_1, x_2) = x_1^4 + x_2^4 - 0.62x_1^2 - 0.62x_2^2$$

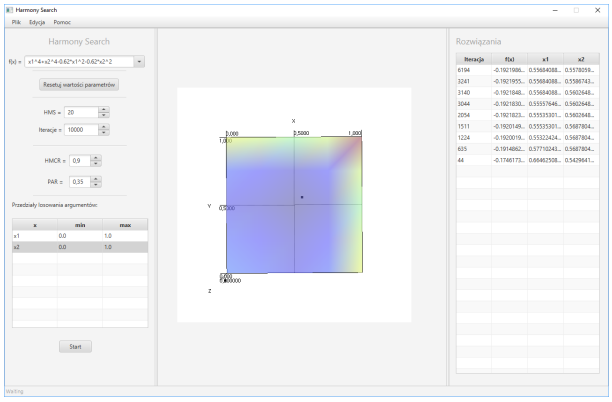
. Rysunek 5 prezentuje trójwymiarowy wykres funkcji na przedziale $x_1, x_2 \in < -1, 1 >$ wraz z zaznaczoną najmniejszą obliczoną wartością $f(-0.56, 56) = -0.19$. Dokładną wartość $10^(- 15)$ można odczytać po prawej stronie w tabeli `Rozwiązania`i.



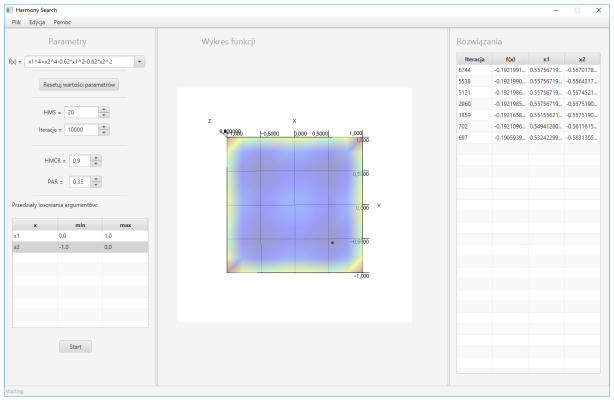
Rysunek 5: Wykres funkcji czterech minimum $x_1, x_2 \in < -1, 1 >$

4.1.1 Zmiana zakresu dla argumentów

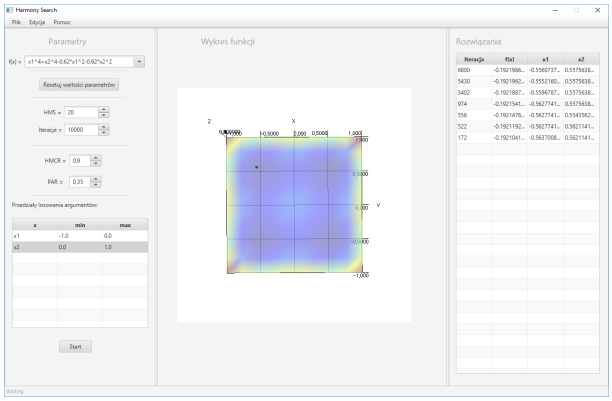
By wyznaczyć wartość interesującego nas minimum należy zawęzić przedział. Przypadki prezentowane na rysunkach 4.1.1, kolejno pokazują wszystkie minima zależnie od podanego przedziału. Jak możemy dostrzec na wykresach algorytm znalazł rozwiązanie na każdym przedziale.



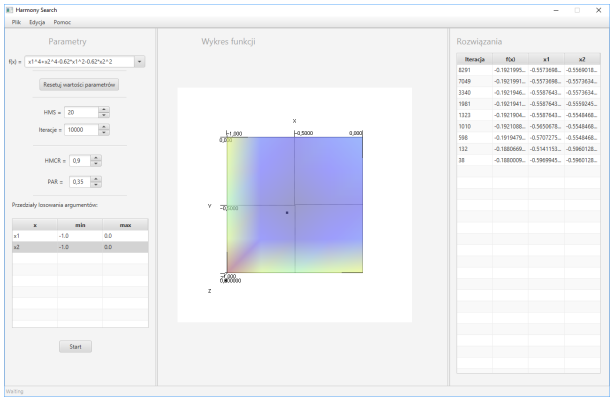
Rysunek 6: $x_1 \in < 0, 1 >, x_2 \in < 0, 1 >$



Rysunek 7: $x_1 \in < 0, 1 >, x_2 \in < -1, 0 >$



Rysunek 8: $x_1 \in < -1, 0 >, x_2 \in < 0, 1 >$



Rysunek 9: $x_1 \in < -1, 0 >, x_2 \in < -1, 0 >$

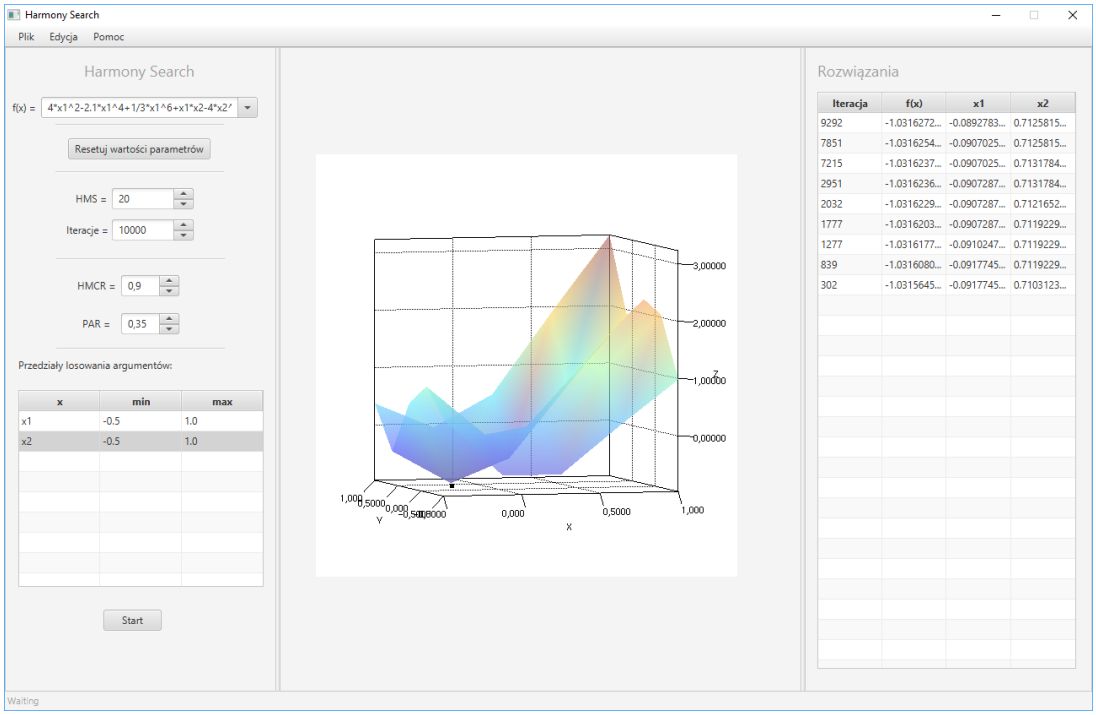
Rysunek 10: Wykres funkcji czterech minimum dla różnych przedziałów

4.2 Funkcja Gemm

Kolejny przykład przedstawiał funkcję opisaną równaniem

$$f(x_1, x_2) = x_1^4 + x_2^4 - 0.62x_1^2 - 0.62x_2^2$$

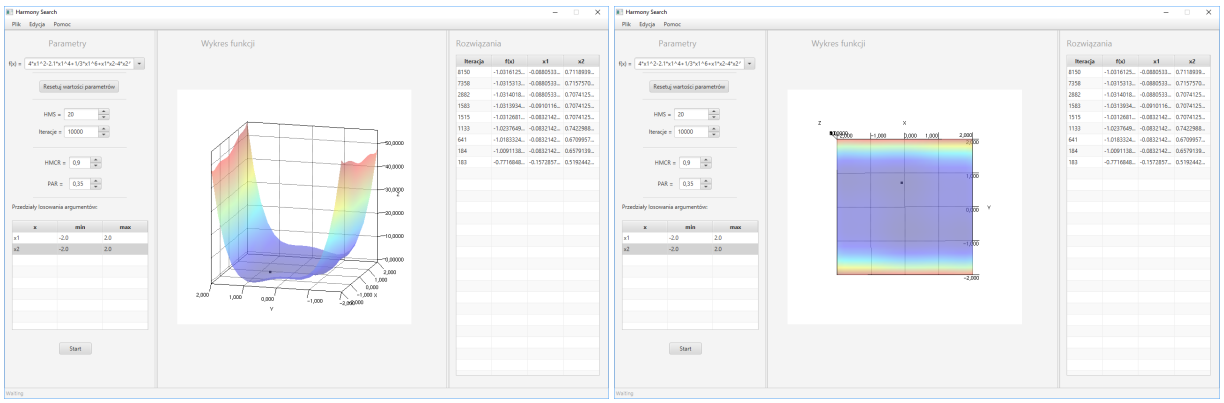
. Początkowo zakres dla argumentów funkcji został przyjęty jako $x_1, x_2 \in < -0.5, 1 >$. Wykres funkcji wraz z rozwiązaniem zaprezentowany jest na 11.



Rysunek 11: Wykres funkcji Gemm dla $x_1, x_2 \in < -0.5, 1 >$

4.2.1 Zmiana zakresu dla argumentów

Jak w poprzednim przykładzie tak i tutaj postanowiono zmienić zakres wyszukiwania argumentów na $x_1, x_2 \in < -2, 2 >$. Porównując oba rozwiązania można stwierdzić, że odnalezione punkty nie różnią się zbytnio.



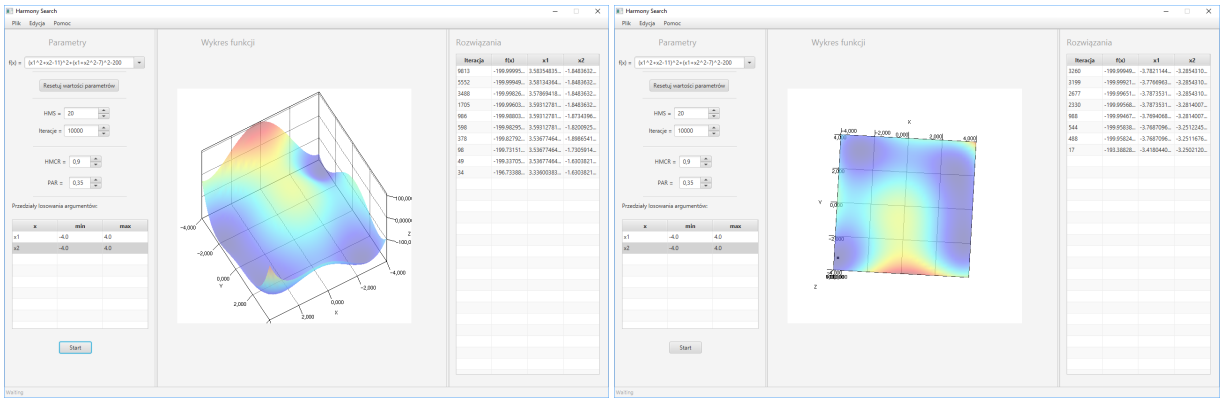
Rysunek 12: Wykresy funkcji Gemm dla $x_1, x_2 \in < -2, 2 >$

4.3 Funkcja Himmelblau

Funkcja Himmelbalu również jak 4.1 posiada cztery minima lokalne zlokalizowane po jednym w każdej ćwiartce na przedziale $x_{1,2} \in (-5, 5)$. Funkcja posiada wzór:

$$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 - 200$$

. Dla pokazania, że funkcja wyszukuje rozwiązania z innego przedziału jak domyślny ograniczono się do przedziału z pierwszej ćwiartki $x_1, x_2 \in (0, 10)$. Na wykresie **??** zaprezentowany został wykres funkcji wraz z naniesionym najlepszym rozwiązaniem.



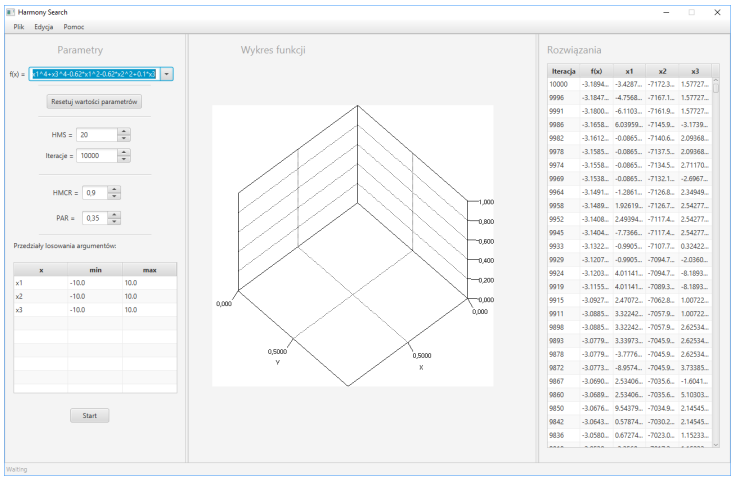
Rysunek 13: Wykresy funkcji Himmelblau dla $x_1, x_2 \in < -4, 4 >$

4.4 Funkcja trzech zmiennych

By przetestować działanie programu i algorytmu dla funkcji z trzema zmiennymi wybrano funkcję

$$f(x_1, x_2, x_3) = x_1^4 + x_3^4 - 0.62x_1^2 - 0.62x_2^2 + 0.1x_3$$

. Oczywistym jest, że wykres czterowymiarowy jest niemożliwy do wyrysowania dlatego okno wykresu zostało puste co prezentuje 14. W tysięcznej iteracji obliczono minimalną wartość funkcji $f(-3.43, -7171, 1.58) = -3.19$. Można się spodziewać, że w kolejnych iteracjach znaleziona by była mniejsza wartość minimalna lecz ograniczenie spowodowało zatrzymacie cyklu obliczeń. W takim przypadku najlepszym rozwiązaniem jest zmniejszenie obszaru dla zmiennych x.



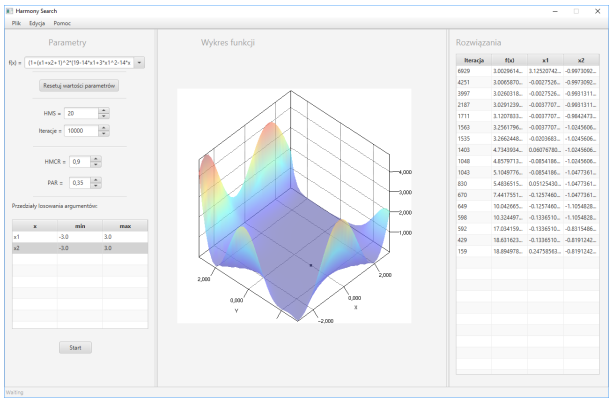
Rysunek 14: Wykres funkcji trzech zmiennych $x_1, x_2, x_3 \in < -10, 10 >$

4.5 Funkcja ceny złota

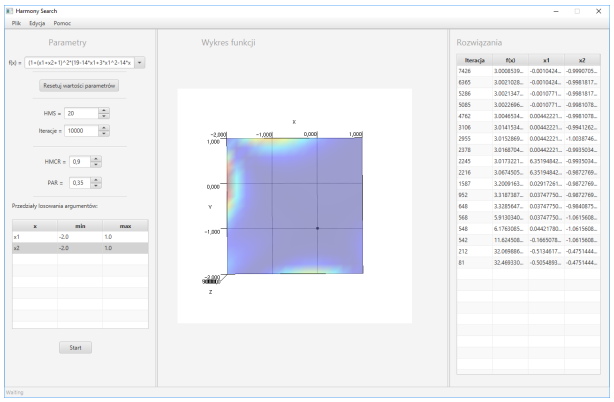
Funkcja Ceny złota [1] posiada kilka minimów lokalnych umiejscowionych blisko siebie. Opisuje ją jeden z dłuższych wzorów

$$f(x_1, x_2) = (1+(x_1+x_2+1)^2*(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2))*(30+(2x_1-3x_2)^2*(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2$$

. Wpierw na rysunku 4.5 ostało przedstawione rozwiązanie dla $x_1, x_2 \in < -3, 3 >$. Następnie na rysunku 4.5 zakres został zmieniony na $x_1, x_2, x_2 \in < -2, 0 >$. Zarówno dla jednego jak i drugiego zakresu program obliczył właściwe rozwiązanie.



Rysunek 15: Wykres ceny złota dla $x_1, x_2 \in < -3, 3 >$



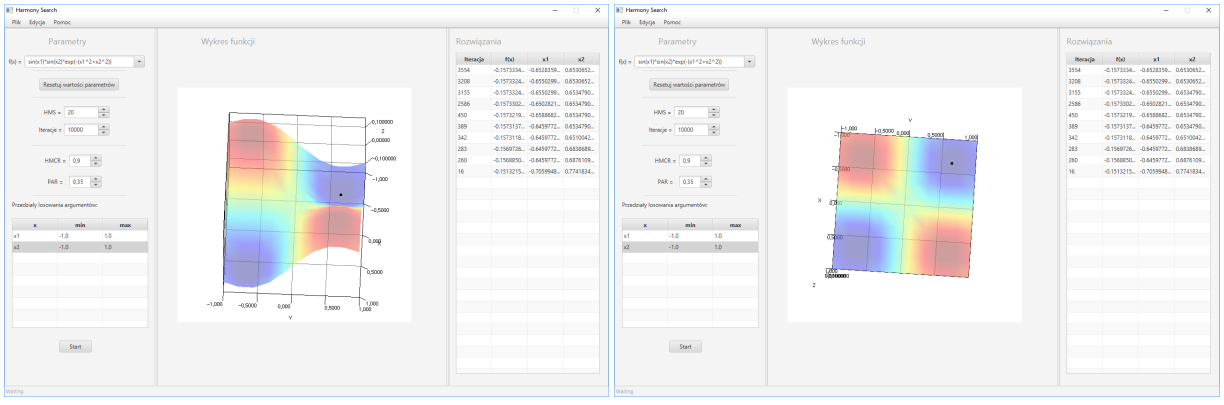
Rysunek 16: Wykres ceny złota dla $x_1, x_2 \in < -2, 0 >$

4.6 Dwuwymiarowa funkcja sinusoidalna

Dwuwymiarowa funkcja sinus posiada bardzo zróżnicowany przebieg oraz wiele równoważnych minimów lokalnych. Wzór funkcji przedstawiony jest równaniem

$$f(x_1, x_2) = \sin x_1 \sin x_2 e^{-(x_1^2+x_2^2)}$$

. Rysunek 4.6 prezentuje wykres funkcji dla argumentów z przedziału $< -1, 1$.



Rysunek 17: Wykres funkcji sinus dla $x_1, x_2 \in < -1, 1 >$

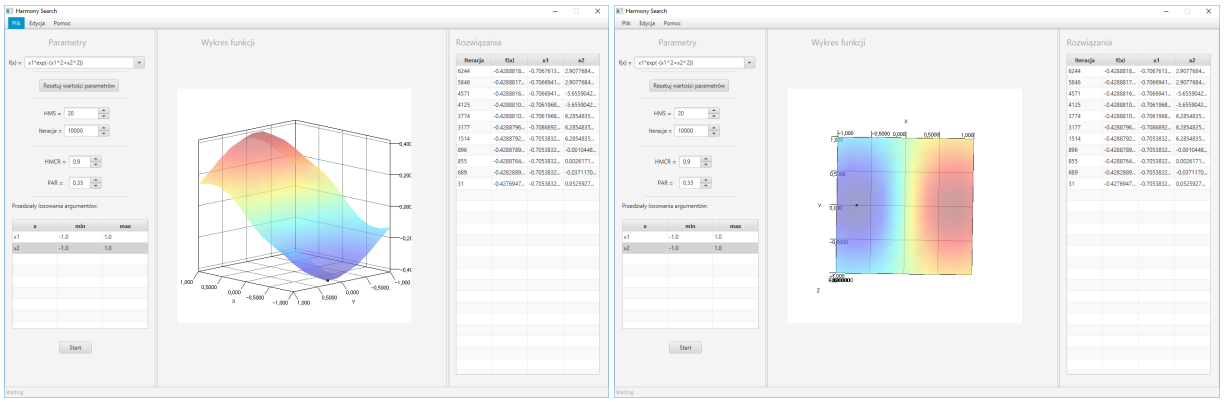
4.7 Dwuwymiarowa funkcja sinusoidalna z eksponentom

Ostatnia przedstawiana funkcja ukazuje przypadek połączenia funkcji sinusa-cosinusa z eksponent. Przypadek jest o tyle ciekawy, że większość wartości funkcji jest dla x_1, x_2 jest do siebie zbliżona. Blisko

jej wartość ulega dużemu odchyłowi. Wzór funkcji przedstawiony jest jako:

$$f(x) = x_1 e^{-(x_1^2+x_2^2)}$$

. Rysunek 4.7 prezentuje działanie programu z obliczoną wartością minimalną. Jak możemy dostrzec program dobrze wyliczył jej wartość co potwierdza jego odporność na działanie funkcji, dla której większość argumentów jest stała.



Rysunek 18: Wykres funkcji sinus z eksponentom $x_1, x_2 \in < -1, 1 >$

4.8 Podsumowanie testów

Sprawdzając działanie algorytmu dla różnych funkcji można stwierdzić, że poprawnie znajduje ich minimum lokalne. Dla funkcji jedno i dwu wymiarowych algorytm działa idealnie. Podczas szukania minimum funkcji trzech zmiennych należy maksymalnie zawęzić obszar. Algorytm działa na tyle szybko, że dla użytkownika nie odczuwalny jest dyskomfort podczas użytkowania programu. Dodatkowo gdy funkcja posiada więcej niż jedno minimum można znaleźć wszystkie z nich zmniejszając przedział dla argumentów. Taka sytuacja została zaprezentowana w punkcie 4.1.1.

5 Problemy podczas pracy

Jednym z problemów z jakim musiał poradzić sobie program było prezentowanie danych. Algorytm jest zdolny do obliczania funkcji wielu zmiennych dla $x_n, n \in N$. Prezentowanie zmiennych w tabelach jest rzeczą dającą się w prosty sposób rozwinąć. Jednak rysowanie wykresu dla $n \leq 2$ ilości zmiennych nie jest możliwy. Najładniejsze wykresy powstają dla $n = 2$ dlatego funkcje o dwóch zmiennych zostały przedstawione w rozdziale 4.

6 Wnioski końcowe

Program obliczający minimum funkcji wielu zmiennych według algorytmu *Harmony Search* działa tak jak założono na początku. Spełniły się domniemania na temat szybkości funkcji. Dodatkowo szybkość poprawiła się gdy zamiast zwykłej tablicy dla `HM` przyjęto `SortedSet`. Oczywistym jest, że program najdokładniej przybliży rozwiązanie gdy zakres wartości x będzie jak najbliższy minimum funkcji. Dodatkowo gdy zmniejszymy ilość iteracji program zakończy szybciej swoje działanie, lecz kosztem gorszego przybliżenia. Gdy wymagane jest obliczenie każdego z minimum funkcji możemy je znaleźć i obliczyć jego wartość o ile znamy przedział, w którym jest ono umiejscowione.

Literatura

[1] Virtual library of simulation experiments. https://www.sfu.ca/ ssurjano/goldpr.html.

[2] Jzy3d. http://www.jzy3d.org/, 2016.

[3] Kowal Andrzej. Algorytm „harmony search".

[4] Mariusz Gromada. Mathparser. https://mvnrepository.com/artifact/org.mariuszgromada.math/MathParser.org-mXparser, 2017.

[5] G.V. Loganathan Kang Seok Lee, Zong Woo Geem. A new heuristic optimization algorithm: Harmony search, 2005.